Test con log resultados

Artillery:

Request http con status ok = 1000

Request rate = 439/sec

Tiempo de respuesta http:

min = 2

max = 61

media = 30.3

Prof:

Ya que el prof es a un nivel bajo y da mucha data, dejo el archivo en la carpeta del proyecto

Autocannon:

```
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/api/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	147 ms	168 ms	233 ms	259 ms	172.86 ms	21.54 ms	307 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	400	400	598	641	577.25	57.91	400
Bytes/Sec	580 kB	580 kB	867 kB	930 kB	837 kB	83.9 kB	580 kB

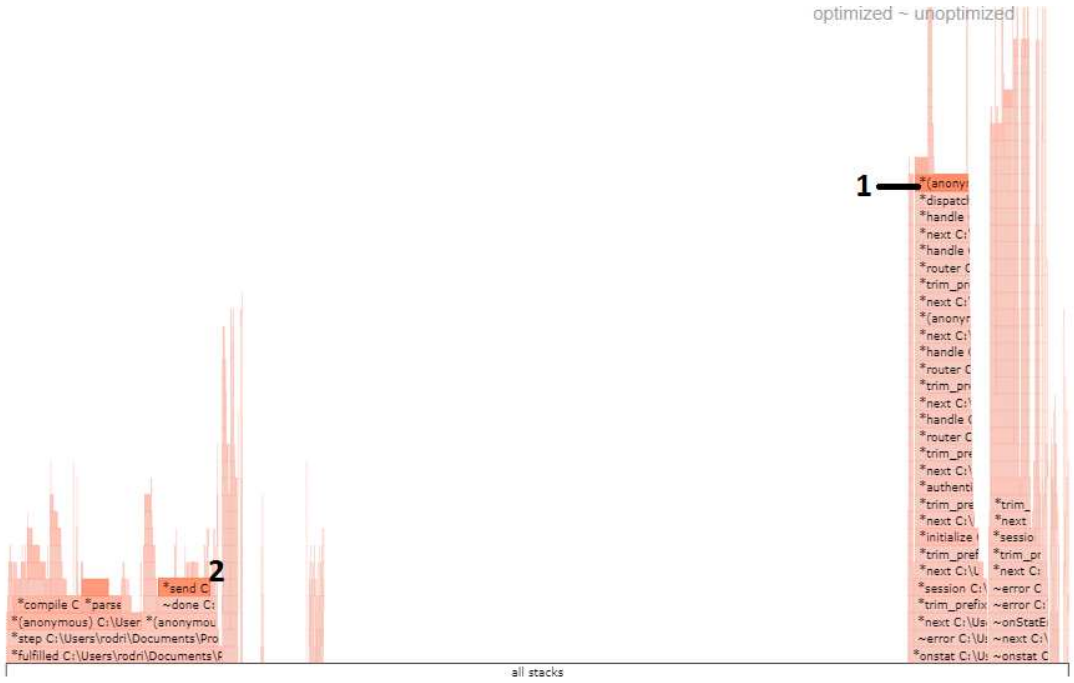
```
Req/Bytes counts sampled once per second.
# of samples: 20

12k requests in 20.14s, 16.7 MB read
```

Inspect:

```
1           import { Router } from "express";
2           import minimist from 'minimist';
3           import os from 'os'
4           //import compression from "compression";
5
6           //import logger from '../logger.js'
7
8           const router = Router();
9
10          router.get('/',  (req, res) => {
11    0.3 ms    const info = {
12    0.6 ms        cpus: os.cpus().length,
13    3.8 ms        arguments: JSON.stringify(minimist(process.argv.slice(2).map((arg, i) => (i, arg)))),
14    0.2 ms        path: `${process.argv[0]}`,
15    0.1 ms        so: process.platform,
16                  processId: process.pid,
17                  nodeVer: process.version,
18                  folder: process.cwd(),
19                  rss: process.memoryUsage().rss,
20              }
21    1.0 ms    console.log(info);
22    2.4 ms    res.render('info', { layout: 'main' , info: info});
23          })
24
25          export default router;
```

0x grafico flama:



1: procesos de ruta info, console.log y recopilacion de data

2: vista handlebars de la ruta

Test sin log resultados

Artillery:

Request http con status ok = 1000

Request rate = 720/sec

Tiempo de respuesta http:

min = 1

max = 23

media = 12.1

Autocannon:

```
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/api/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	77 ms	87 ms	108 ms	124 ms	89.13 ms	9.23 ms	184 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	800	800	1146	1199	1118.7	85.18	800
Bytes/Sec	1.14 MB	1.14 MB	1.64 MB	1.71 MB	1.6 MB	122 kB	1.14 MB

```
Req/Bytes counts sampled once per second.
# of samples: 20

22k requests in 20.12s, 31.9 MB read
```

Inspect:

```
1                import { Router } from "express";
2                import minimist from 'minimist';
3                import os from 'os'
4                //import compression from "compression";
5
6                //import logger from '../logger.js'
7
8                const router = Router();
9
10               router.get('/',  (req, res) => {
11      0.1 ms       const info = {
12      0.4 ms           cpus: os.cpus().length,
13      2.4 ms           arguments: JSON.stringify(minimist(process.argv.slice(2).map((arg, i) => (i, arg)))),
14      0.2 ms           path: `${process.argv[0]}`,
15                       so: process.platform,
16                       processId: process.pid,
17                       nodeVer: process.version,
18                       folder: process.cwd(),
19      0.2 ms           rss: process.memoryUsage().rss,
20                   }
21      1.4 ms       res.render('info', { layout: 'main' , info: info});
22               })
23
24               export default router;
```

0x grafico flama:

Se ve la misma carga que en el test con el log, pero en este caso, los procesos de la ruta info cuentan con menos carga.

Conclusion:

Entre los resultados que ya esperaba me encontre con las siguientes diferencias.

Artillery demostro que:

* Sin el log pasa de request rate de 439/sec a 720/sec

* Sin el log los tiempos de respuesta bajan considerablemente:

- min = con log: 2 sin log: 1 diferencia de: 1

- max = con log: 61 sin log: 23 diferencia de: 38

- media = con log: 30.3 sin log: 12.1 diferencia de: 18.2

Autocannon demostro que:

* Sin el log el promedio de la latencia se reduce considerablemente

obteniendo 172.86 ms con el log y 89.13 ms sin el log obteniendo una ganancia de 83.56 ms.

Inspect demostro que:

* La lectura de argumentos y el renderizado del front consumen muchos recursos.

* Quitando el log se agiliza el tiempo de respuesta.

Entre los resultados que no esperaba, gracias a el grafico de flama y el inspect encontre que enviar las vistas desde el backend consume muchos recursos.

Gracias a estos datos puedo entender que es importante remover el codigo redundante y que no servira para el usuario antes de lanzar el server a produccion, tanto logs como funciones que pueden ser simplificadas, tambien puede entender un poco mas la importancia de separar el front end del backend ya que enviar las vistas desde el back consume muchos recursos, realentizando asi el funcionamiento principal del servidor.

Sin usar gzip la ruta /api/info transfiere 1.4 kB

Usando gzip la ruta /api/info transfiere 884 B

- Rodrigo Vergara